Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)
Кафедра компьютерных систем в управлении и проектировании (КСУП)

СОЗДАНИЕ ПРИЛОЖЕНИЯ В MICROSOFT VISUAL STUDIO ДЛЯ РАБОТЫ СО СВЯЗАННЫМИ ТАБЛИЦАМИ БАЗЫ ДАННЫХ

Отчет по лабораторной работе по дисциплине «Основы разработки баз данных»

Студент гр. 573-3:

_____ Р.В. Слиньков
дата

_____
подпись

Руководитель:

Преподаватель:

_____ _____ Р. О. Остапенко
оценка подпись

_____
дата

Томск, 2025

Введение

В современном мире базы данных играют ключевую роль в системах хранения и обработки информации. Они используются в самых разных областях — от небольших веб-приложений до крупных корпоративных систем. Эффективная организация данных позволяет оптимизировать работу программных продуктов, обеспечить целостность и безопасность информации, а также упростить её поиск и анализ.

Цель данной лабораторной работы:

Изучение основных особенностей создания приложения для работы со связанными таблицами базы данных в Microsoft Visual Studio для своей предметной области (на примере базы данных для кафе).

Задачи:

Реализовать программу согласно требованиям:

1. Создать форму/формы проекта для работы с данными по связанным таблицам. Обязательно 1:M на одной форме, M:M через 1:M и M:1 на одной форме;
2. Реализовать шаблон проектирования «Одиночка» (англ. Singleton);
3. Обработать исключения в программе, возникающие при некорректной работе с данными (не менее 2-3 разных ошибок); Сформировать конфигурационный файл для изменения пути к БД в приложении;
4. Реализовать поиск, фильтрацию данных;
5. Создать вычисляемые колонки (не менее 1-2 разных);
6. Создать подстановочные колонки (не менее 1-2 разных);
7. Использовать различные элементы управления Label, TextBox, Button, DataGridView, BindingNavigator и др.;
8. Настроить для удобства BindingNavigator (всплывающие подсказки обязательно);
9. Протестировать работу полученного приложения.

1 ОСНОВНАЯ ЧАСТЬ

1.1 Описание архитектуры для создания приложения и ход выполнения работы

Windows Forms — это технология для создания настольных приложений на платформе .NET, которая предоставляет мощный инструментарий для разработки пользовательского интерфейса. Она позволяет использовать широкий набор элементов управления, таких как Label для отображения текста, TextBox для ввода данных, ComboBox для выбора из списка, CheckBox для выбора опций, PictureBox для работы с изображениями, DataGridView для отображения данных в табличном виде и BindingNavigator для навигации по записям. Эти элементы обеспечивают гибкость и удобство при создании интерактивных приложений.

В рамках прошлой лабораторной работы была реализована архитектура приложения, основанная на принципах разделения ответственности и модульности. Для взаимодействия с базой данных использовался **паттерн репозитория**, который представляет собой прослойку между бизнес-логикой приложения и уровнем доступа к данным. Этот паттерн реализован через интерфейсы (IWarehouseRepository, ISupplierRepository, ISupplyRepository, IStorageZoneRepository, IEmployeeRepository) и их конкретные реализации в проекте Infrastructure. Каждый репозиторий предоставляет методы для выполнения операций CRUD (Create, Read, Update, Delete) над соответствующей сущностью, например, GetAll, Add, Update, Delete.

В данной лабораторной работе, были реализованы в полной мере связи 1:M и решены связи M:M через 1:M – M:1.

Ход выполнения заданий

1. **Создание формы/форм для работы с данными по связанным таблицам:**

Была обновлена форма ProductAccounting. На форме реализованы связи:

1. 1:M: один сотрудник связан с несколькими записями учёта (через таблицу product_accounting и employee).

2. M:M через 1:M: поставки (supply) связаны с продукцией и поставщиками через таблицу product_accounting (связь M:M реализована через 1:M между supply и product_accounting).

3. M:1: несколько записей учёта ссылаются на одну зону хранения (storage_zone).

2. **Реализация шаблона «Одиночка»:**

Шаблон проектирования «Одиночка» (Singleton) уже был реализован в рамках предыдущей лабораторной работы в классе AppContext. Этот класс предоставляет единую точку доступа к репозиториям через статическое свойство Instance, что позволяет централизованно управлять зависимостями в приложении. В текущей работе использование Singleton продолжило обеспечивать удобство работы с репозиториями, такими как IProductAccountingRepository, ISupplyRepository и другими.

3. **Обработка исключений:**

Были обработаны ошибки, возникающие с корректным вводом значений (ПРИЛОЖЕНИЕ А Листинг 1.1), ошибки, возникающие с поиском/фильтрац ией данных (ПРИЛОЖЕНИЕ А Листинг 1.2).

4. **Реализация поиска и фильтрации данных:**

Добавлены функции поиска и фильтрации данных на формах. Поиск реализован через текстовое поле в BindingNavigator, а фильтрация — через CheckBox с динамическим обновлением данных (ПРИЛОЖЕНИЕ А Листинг 1.3).

5. **Создание вычисляемых колонок:**

В DataGridView на форме ProductAccountingForm были добавлены две вычисляемые колонки для автоматического расчёта значений на основе данных (ПРИЛОЖЕНИЕ А Листинг 1.4):

- **DaysSinceAccounting**: Колонка отображает количество дней, прошедших с даты учёта (AccountingDate). Значение рассчитывается как

разница между текущей датой (DateTime.Now) и AccountingDate, что позволяет пользователю быстро оценить, насколько давно была произведена запись. Например, если запись была сделана 10 дней назад, в колонке отобразится значение "10".

- **IsRecentMovement**: Колонка представляет собой флаг ("Да" или "Нет"), указывающий, было ли движение продукции за последние 30 дней. Значение определяется на основе поля LastMovementDate: если с даты последнего движения прошло менее 30 дней, отображается "Да", иначе — "Нет".

6. **Создание подстановочных колонок:**

Добавлена одна подстановочная колонка в DataGridView в ProductAccounting под названием MovementStatus с выподающим списком (ПРИЛОЖЕНИЕ А Листинг 1.5). Эта колонка позволяет пользователю выбирать одно из трёх значений: "В наличии", "Перемещено" или "Списано". Реализация включала создание колонки типа DataGridViewComboBoxColumn, настройку её источника данных (списка статусов) и привязку к полю MovementStatus в модели данных. Были добавлены обработчики событий CellValueChanged и CurrentCellDirtyStateChanged для немедленного сохранения изменений в базе данных при выборе нового статуса.

7. **Настройка BindingNavigator:**

BindingNavigator на всех формах был дополнен элементами управления для повышения удобства работы с данными (ПРИЛОЖЕНИЕ А Листинг 1.6). Были добавлены:

- Строка поиска (ToolStripTextBox): Позволяет пользователю вводить запрос для поиска записей.
- Кнопка "Поиск" (ToolStripButton): Запускает поиск по введённому запросу.
- Кнопка "Сброс" (ToolStripButton): Очищает строку поиска и сбрасывает фильтр, отображая все записи.

- **CheckBox** фильтрации: Включает или отключает фильтрацию данных.

Каждый элемент снабжён всплывающей подсказкой, поясняющей его назначение (например, "Введите текст для поиска" для строки поиска).

1.2 Результаты выполнения

В результате выполнения лабораторной работы было улучшено Windows Forms приложение "Система складского учёта", соответствующее требованиям лабораторной работы. Приложение позволяет управлять данными о складах, поставщиках, поставках, зонах хранения и сотрудниках. Реализованы следующие функции:

Связи 1:M и M:M (1:M – M:1). Форма ProductAccountingForm была дополнена поддержкой связей между таблицами базы данных, что позволило отображать данные из связанных сущностей (сотрудники, поставки, зоны хранения) в едином интерфейсе. Реализация связей 1:M (например, один сотрудник отвечает за несколько записей учёта) и M:1 (несколько записей учёта ссылаются на одну зону хранения) обеспечила удобное отображение данных в DataGridView. Связь M:M между поставками и продукцией реализована через промежуточную таблицу **supply**, что позволило корректно отображать данные о поставках и продукции, связанных через записи учёта. Это улучшило наглядность и упростило работу с данными для пользователя (рисунок 1.1).

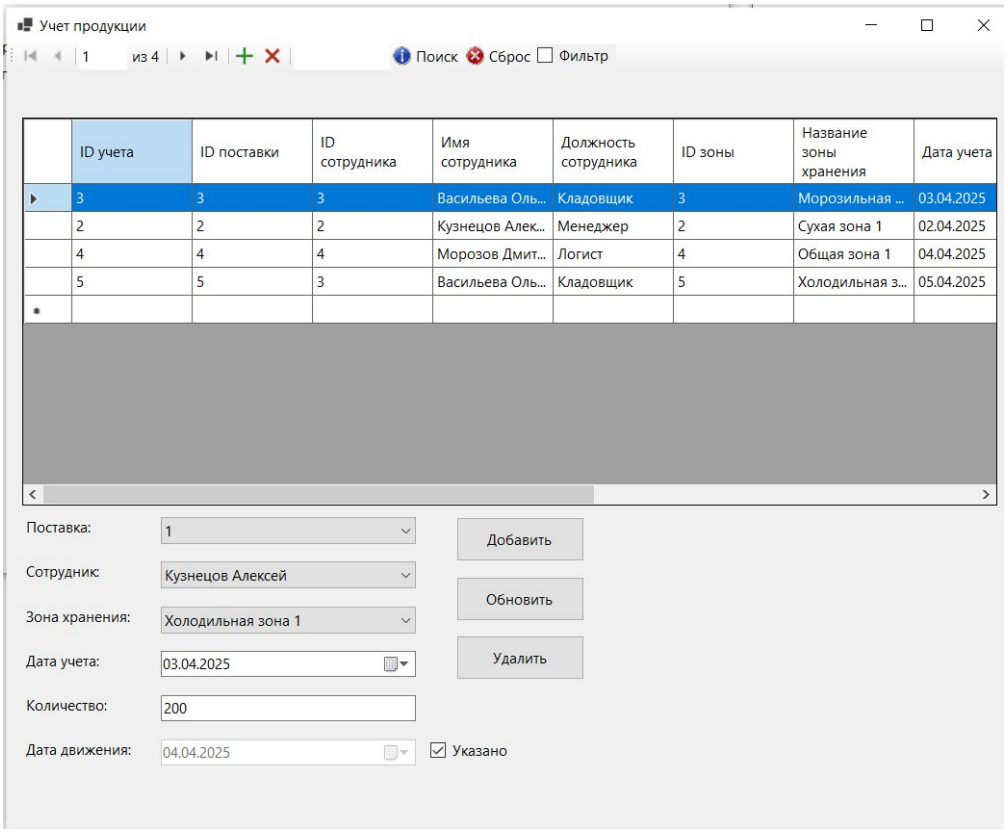

Рисунок 1.1 – Связи (1:M – M:1) в учете товаров

Обработка исключений. В приложение добавлена обработка исключений, что повысило его надёжность и устойчивость к ошибкам. Например, при некорректном вводе данных, таком как отрицательное количество продукции, пользователю отображается информативное сообщение об ошибке (рисунок 1.2), что помогает избежать некорректных операций. Аналогично, при попытке поиска или фильтрации данных, которые отсутствуют в базе, приложение корректно обрабатывает такую ситуацию, уведомляя пользователя (рисунок 1.3). Это позволяет пользователю быстро понять причину проблемы и предпринять необходимые действия, улучшая общий опыт работы с приложением (рисунок 1.2 – 1.3).

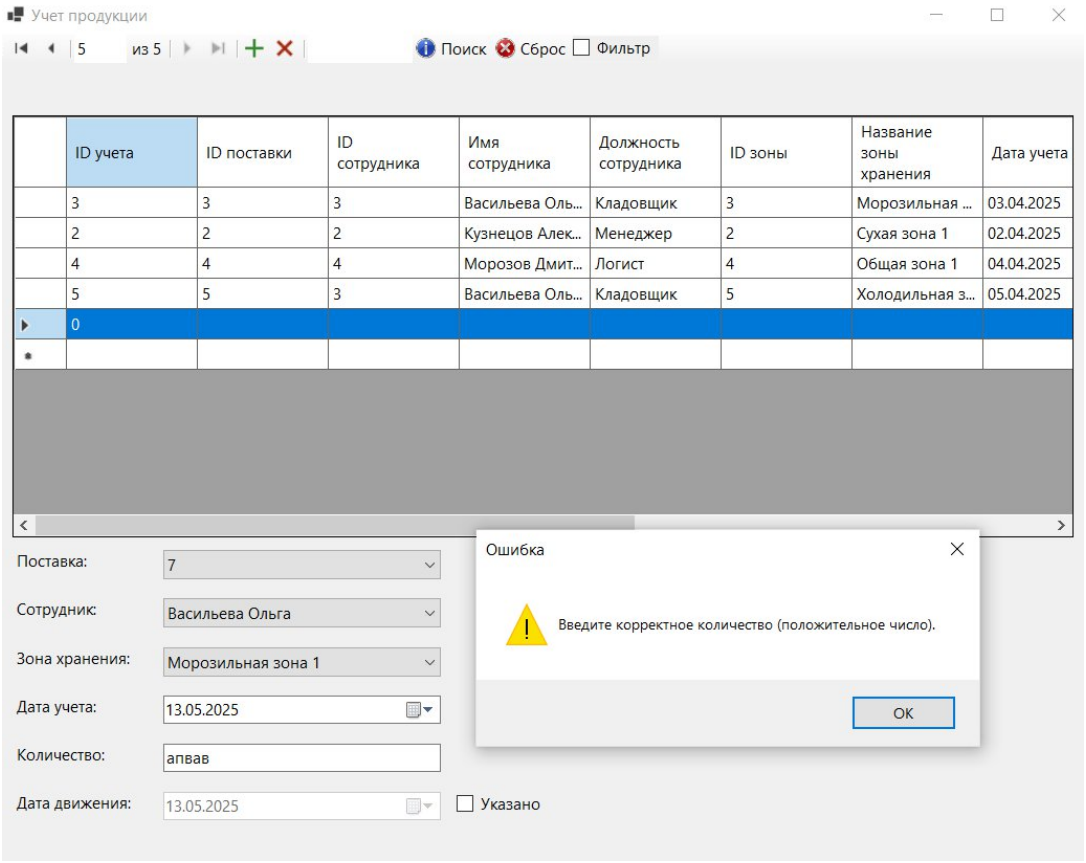

Рисунок 1.2 – Обработка исключения связанного с некорректным вводом

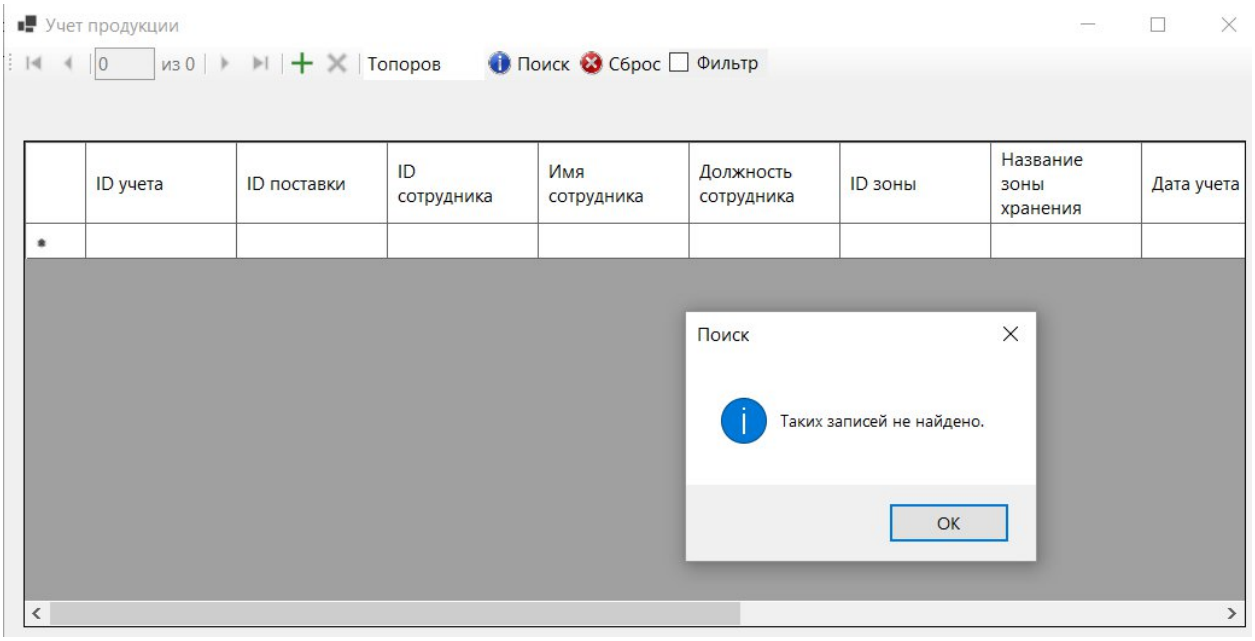

Рисунок 1.3 – Обработка исключения связанного с поиском данных

Поиск и фильтрация данных. Функциональность поиска и фильтрации данных реализована через BindingNavigator, что делает её максимально удобной для пользователя. Пользователь может ввести запрос в

текстовое поле и нажать кнопку "Поиск", чтобы быстро найти нужные записи. Фильтрация активируется через CheckBox, что позволяет динамически обновлять отображаемые данные без необходимости перезапуска формы. Эта функция особенно полезна при работе с большим количеством записей, так как позволяет быстро находить нужную информацию, экономя время пользователя (рисунок 1.4).

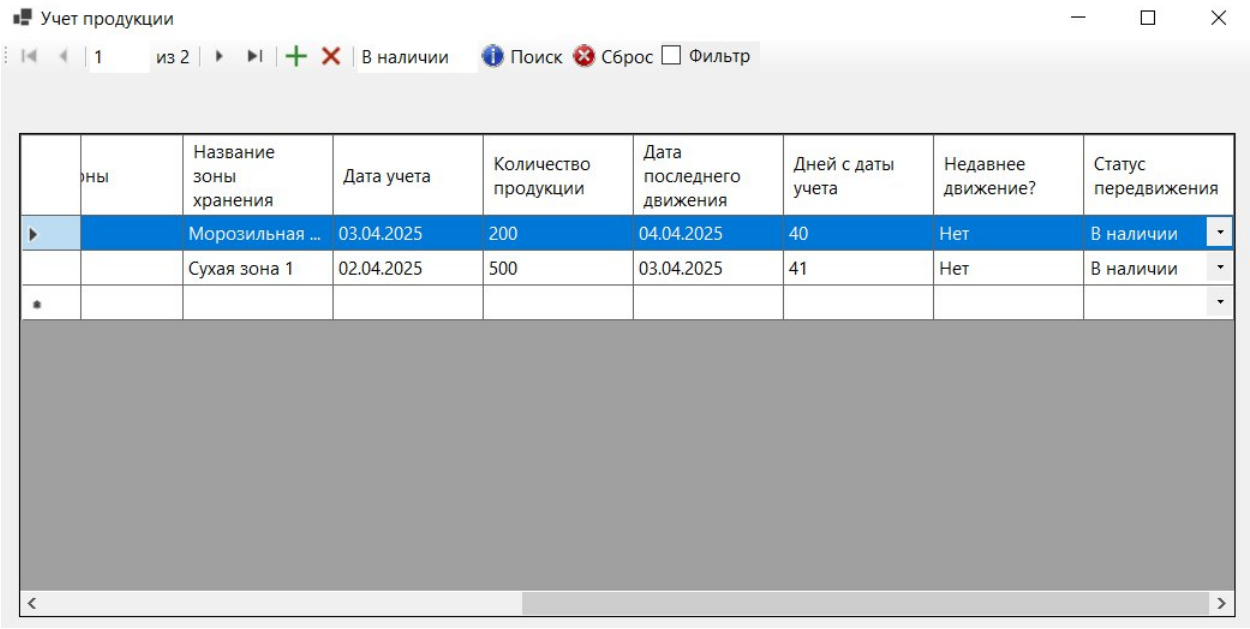

Рисунок 1.4 – Поиск и фильтрация данных

Вычисляемые колонки в «Учете товаров» «Дней с даты учета» и «Недавнее передвижение». В DataGridView на форме ProductAccountingForm добавлены две вычисляемые колонки, которые автоматически рассчитывают значения на основе данных. Колонка «Дней с даты учёта» (DaysSinceAccounting) отображает количество дней, прошедших с даты учёта, что помогает пользователю быстро оценить, насколько давно была произведена запись. Колонка «Недавнее передвижение» (IsRecentMovement) показывает, было ли движение продукции за последние 30 дней, отображая значение "Да" или "Нет". Эти колонки делают данные более информативными, позволяя пользователю принимать решения без необходимости дополнительных расчётов. Например, пользователь может сразу увидеть,

какие записи требуют внимания из-за длительного отсутствия движения (рисунок 1.5).

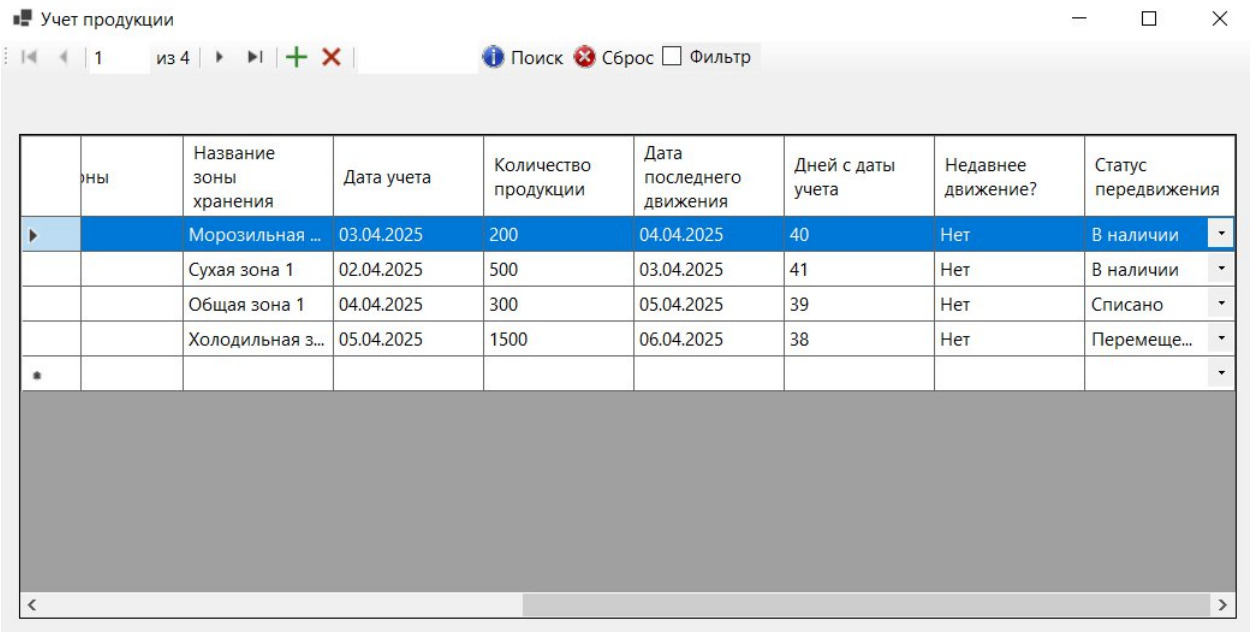

Рисунок 1.5 – Вычисляемые колонки

Подстановочная колонка в «Учете товаров» «Статус передвижения». Добавлена подстановочная колонка «Статус передвижения» (MovementStatus) с выпадающим списком, который позволяет пользователю выбирать одно из трёх значений: "В наличии", "Перемещено" или "Списано". Это улучшает контроль над статусом продукции, так как пользователь может быстро обновить статус прямо в DataGridView, без необходимости открывать отдельные формы или вводить данные вручную. Выпадающий список также исключает возможность ввода некорректных значений, что повышает целостность данных (рисунок 1.6 – 1.7).

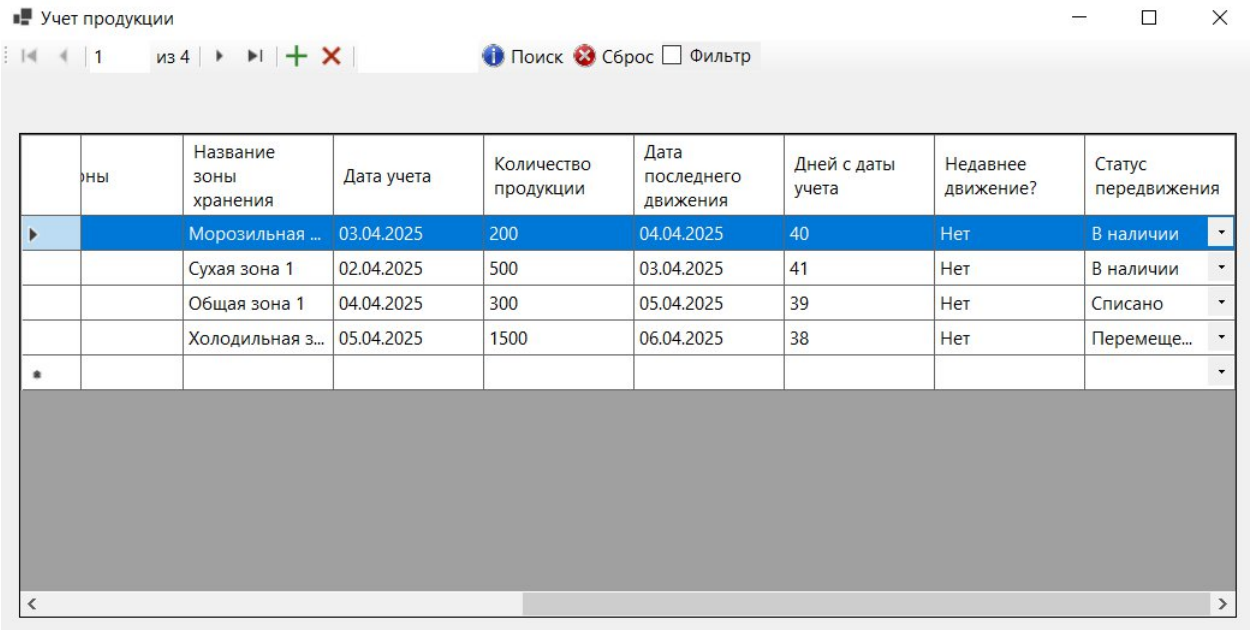

Рисунок 1.6 – Подстановочная колонка

```
Ссылок: 1
private void DataGridViewProductAccountings_CurrentCellDirtyStateChanged(object sender, EventArgs e)
{
    if (dataGridViewProductAccountings.IsCurrentCellDirty)
    {
        dataGridViewProductAccountings.CommitEdit(DataGridViewDataErrorContexts.Commit);
    }
}

Ссылок: 1
private void DataGridViewProductAccountings_CellValueChanged(object sender, DataGridViewCellEventArgs e)
{
    if (e.ColumnIndex == dataGridViewProductAccountings.Columns["MovementStatus"].Index && e.RowIndex >= 0)
    {
        try
        {
            var row = dataGridViewProductAccountings.Rows[e.RowIndex];
            var productAccId = (int)row.Cells["ProductAccId"].Value;
            var newStatus = row.Cells["MovementStatus"].Value?.ToString();

            var productAccounting = _productAccountingRepository.GetById(productAccId);
            if (productAccounting != null)
            {
                productAccounting.MovementStatus = newStatus;
                _productAccountingRepository.Update(productAccounting);
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Ошибка обновления статуса: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            LoadProductAccountings();
        }
    }
}
```

Рисунок 1.7 – Код методов для подстановочной колонки

Настройка BindingNavigator для поиска и фильтрации. BindingNavigator был дополнен элементами управления для поиска и фильтрации: строка поиска, кнопка "Поиск", кнопка "Сброс" и CheckBox для фильтрации. Каждая кнопка снабжена всплывающей подсказкой, что делает интерфейс более интуитивным даже для неопытных пользователей. Например,

подсказка на кнопке "Поиск" сообщает: "Найти записи по введённому запросу", а на кнопке "Сброс" — "Очистить строку поиска и показать все записи". Эти улучшения упрощают навигацию по данным и делают работу с приложением более комфортной (рисунок 1.8)

Рисунок 1.8 – Настройка BindingNavigator

Заключение

В ходе выполнения лабораторной работы было улучшено Windows Forms приложение "Система складского учёта", которое полностью соответствует требованиям лабораторной работы №3. Приложение предоставляет удобный пользовательский интерфейс для управления данными об учёте продукции, сотрудниках, поставках и зонах хранения, обеспечивая выполнение операций просмотра, добавления, редактирования и удаления записей, а также их поиск и фильтрацию.

Ключевые результаты работы:

• Реализованы все пункты лабораторной работы, включая создание формы ProductAccountingForm с поддержкой связей 1:M, M:M через 1:M и M:1, обработку исключений, настройку поиска и фильтрации данных, создание вычисляемых и подстановочных колонок, использование различных элементов управления (Label, TextBox, Button, DataGridView, BindingNavigator), настройку BindingNavigator с всплывающими подсказками и тестирование приложения.

• Архитектура приложения, построенная на паттерне репозитория, обеспечила разделение бизнес-логики и доступа к данным, что повысило модульность и тестируемость кода. Паттерн Singleton, реализованный в классе AppContext в рамках предыдущей работы, продолжил обеспечивать централизованное управление зависимостями.

• Добавлены вычисляемые колонки DaysSinceAccounting и IsRecentMovement, а также подстановочная колонка MovementStatus с выпадающим списком, что соответствует требованиям пунктов 3.2.5 и 3.2.6.

Приложение демонстрирует высокую степень модульности и гибкости благодаря разделению на проекты (Domain, Infrastructure, UI, Application). Использование интерфейсов и паттернов проектирования упрощает дальнейшую поддержку и расширение функциональности, например,

добавление новых форм для управления другими сущностями или интеграцию с другими технологиями баз данных.

ПРИЛОЖЕНИЕ А

Листинг 1.1

```
private bool ValidateInput()
{
    if (cmbSupply.SelectedValue == null)
    {
        MessageBox.Show("Выберите поставку.", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return false;
    }

    if (cmbEmployee.SelectedValue == null)
    {
        MessageBox.Show("Выберите сотрудника.", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return false;
    }

    if (cmbStorageZone.SelectedValue == null)
    {
        MessageBox.Show("Выберите зону хранения.", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return false;
    }

    if (string.IsNullOrWhiteSpace(txtQuantity.Text)
|| !int.TryParse(txtQuantity.Text, out int quantity) || quantity
<= 0)
    {
        MessageBox.Show("Введите корректное количество
(положительное число).", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
        return false;
    }
```

```
    return true;
}
```

Листинг 1.2

```
if (_bindingSource.Count == 0)
{
    MessageBox.Show("Таких записей не найдено.", "Поиск",
MessageBoxButtons.OK, MessageBoxIcon.Information);
    LoadProductAccountings();
}
```

Листинг 1.3

ProductForm.cs

```
private void ApplyFilter(string searchText)
{
    try
    {
        var filteredProductAccountings =
_productAccountingRepository.GetFiltered(searchText);
        var enrichedData = filteredProductAccountings.Select(pa
=>
        {
            var employee =
_employeeRepository.GetById(pa.EmployeeId);
            var lastMovementDate =
pa.LastMovementDate.HasValue ?
DateTime.Parse(pa.LastMovementDate.Value.ToString("dd.MM.yyyy"))
: (DateTime?)null;
            return new ProductAccountingView
            {
                ProductAccId = pa.ProductAccId,
                SupplyId = pa.SupplyId.ToString(),
                EmployeeId = pa.EmployeeId.ToString(),
                EmployeeName = employee?.FullName ?? "Не
найдено",
                EmployeePosition = employee?.Position ?? "Не
указано",
```

```
                StorageId = pa.StorageId.ToString(),
                StorageZone =
_storageZoneRepository.GetById(pa.StorageId)?.ZoneName ?? "Не
найдено",
                AccountingDate = pa.AccountingDate,
                Quantity = pa.Quantity,
                LastMovementDate =
pa.LastMovementDate?.ToString("dd.MM.yyyy") ?? "Не указано",
                DaysSinceAccounting = (DateTime.Now -
pa.AccountingDate).Days,
                IsRecentMovement = lastMovementDate.HasValue &&
(DateTime.Now - lastMovementDate.Value).Days <= 30 ? "Да" :
"Нет",
                MovementStatus = pa.MovementStatus ?? "В
наличии"
            };
        }).ToList();

        _bindingSource.DataSource = new
BindingList<ProductAccountingView>(enrichedData);
        dataGridViewProductAccountings.DataSource =
_bindingSource;

        if (_bindingSource.Count == 0)
        {
            MessageBox.Show("Таких записей не найдено.",
"Поиск", MessageBoxButtons.OK, MessageBoxIcon.Information);
            LoadProductAccountings();
        }
        else
        {
            _bindingSource.Position = 0;
        }
    }
    catch (Exception ex)
    {
```

```
        MessageBox.Show($"Ошибка фильтрации: {ex.Message}",
"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        LoadProductAccountings();
    }
}

private void ToolStripButtonFind_Click(object sender, EventArgs
e)
{
    if (_toolStripTextBoxFind == null)
    {
        MessageBox.Show("Ошибка: toolStripTextBoxFind не
инициализирован.");
        return;
    }

    string searchText = _toolStripTextBoxFind.Text.Trim();
    if (string.IsNullOrWhiteSpace(searchText))
    {
        MessageBox.Show("Введите текст для поиска.", "Внимание",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        return;
    }

    ApplyFilter(searchText);
}

private void ToolStripButtonReset_Click(object sender, EventArgs
e)
{
    LoadProductAccountings();
    _toolStripTextBoxFind.Text = null;
}
```

```
private void CheckBoxFind_CheckedChanged(object sender,
EventArgs e)
{
    if (_toolStripTextBoxFind == null || _checkBoxFind == null)
    {
        MessageBox.Show("Ошибка: элементы не
инициализированы.");
        _checkBoxFind.Checked = false;
        return;
    }

    if (_checkBoxFind.Checked)
    {
        string searchText = _toolStripTextBoxFind.Text.Trim();
        if (string.IsNullOrWhiteSpace(searchText))
        {
            MessageBox.Show("Введите текст для фильтрации.",
"Внимание", MessageBoxButtons.OK, MessageBoxIcon.Information);
            _checkBoxFind.Checked = false;
            return;
        }

        ApplyFilter(searchText);
    }
    else
    {
        LoadProductAccountings();
    }
}
```

ProductRepository.cs

```
public List<ProductAccounting> GetFiltered(string searchText)
{
    var productAccountings = new List<ProductAccounting>();
    using (var conn = _dbConnection.GetConnection())
```

```
    {
        conn.Open();
        string query = @"
            SELECT pa.productAcc_id, pa.supply_id,
pa.employee_id, pa.storage_id, pa.accounting_date, pa.quantity,
pa.last_movement_date, pa.movement_status
            FROM product_accounting pa
            LEFT JOIN employee e ON pa.employee_id =
e.employee_id
            LEFT JOIN storage_zone sz ON pa.storage_id =
sz.storage_id
            WHERE pa.productAcc_id::text ILIKE @search
            OR pa.supply_id::text ILIKE @search
            OR pa.employee_id::text ILIKE @search
            OR e.full_name ILIKE @search
            OR e.position ILIKE @search
            OR pa.storage_id::text ILIKE @search
            OR sz.zone_name ILIKE @search
            OR pa.accounting_date::text ILIKE @search
            OR pa.quantity::text ILIKE @search
            OR pa.movement_status ILIKE @search
            OR (pa.last_movement_date::text ILIKE @search OR
pa.last_movement_date IS NULL)";
        using (var cmd = new NpgsqlCommand(query, conn))
        {
            cmd.Parameters.AddWithValue("search",
$"%{searchText}%");
            using (var reader = cmd.ExecuteReader())
            {
                while (reader.Read())
                {
                    productAccountings.Add(new ProductAccounting
                    {
                        ProductAccId = reader.GetInt32(0),
                        SupplyId = reader.GetInt32(1),
```

```
                        EmployeeId = reader.GetInt32(2),
                        StorageId = reader.GetInt32(3),
                        AccountingDate = reader.GetDateTime(4),
                        Quantity = reader.GetInt32(5),
                        LastMovementDate = reader.IsDBNull(6) ?
null : reader.GetDateTime(6),
                        MovementStatus = reader.IsDBNull(7) ? "В
наличии" : reader.GetString(7)
                    });
                }
            }
        }
    }
    return productAccountings;
}
```

Листинг 1.4

```
public class ProductAccountingView
{
    public int ProductAccId { get; set; }
    public string SupplyId { get; set; }
    public string EmployeeId { get; set; }
    public string EmployeeName { get; set; }
    public string EmployeePosition { get; set; } //
Подстановочная колонка
    public string StorageId { get; set; }
    public string StorageZone { get; set; }
    public DateTime AccountingDate { get; set; }
    public int Quantity { get; set; }
    public string LastMovementDate { get; set; }
    public int DaysSinceAccounting { get; set; } // Вычисляемая
колонка
    public string IsRecentMovement { get; set; } // Вычисляемая
колонка
    public string MovementStatus { get; set; } // Новое поле для
выпадающего списка
```

```
    public override string ToString()
    {
        return $"{ProductAccId} - {EmployeeName} -
{StorageZone}";
    }
}

dataGridViewProductAccountings.Columns.Add(new
DataGridViewTextBoxColumn
{
    DataPropertyName = "DaysSinceAccounting",
    HeaderText = "Дней с даты учета",
    Name = "DaysSinceAccounting",
    ReadOnly = true
});
dataGridViewProductAccountings.Columns.Add(new
DataGridViewTextBoxColumn
{
    DataPropertyName = "IsRecentMovement",
    HeaderText = "Недавнее движение?",
    Name = "IsRecentMovement",
    ReadOnly = true
});

dataGridViewProductAccountings.CellValueChanged +=
DataGridViewProductAccountings_CellValueChanged;
dataGridViewProductAccountings.CurrentCellDirtyStateChanged +=
DataGridViewProductAccountings_CurrentCellDirtyStateChanged;
```

Листинг 1.5

```
// Добавляем выпадающий список для MovementStatus
var comboBoxColumn = new DataGridViewComboBoxColumn
{
    DataPropertyName = "MovementStatus",
    HeaderText = "Статус передвижения",
    Name = "MovementStatus",
```

```
    DataSource = _movementStatuses,
    FlatStyle = FlatStyle.Flat
};
dataGridViewProductAccountings.Columns.Add(comboBoxColumn);
```

Листинг 1.6

```
_bindingSource = new BindingSource();
_movementStatuses = new List<string> { "В наличии",
"Перемещено", "Списано" };


_bindingNavigator = new BindingNavigator(_bindingSource);
_bindingNavigator.Dock = DockStyle.Top;
this.Controls.Add(_bindingNavigator);


_bindingNavigator.MoveFirstItem.ToolTipText = "Перейти к первой
записи";
_bindingNavigator.MovePreviousItem.ToolTipText = "Перейти к
предыдущей записи";
_bindingNavigator.MoveNextItem.ToolTipText = "Перейти к
следующей записи";
_bindingNavigator.MoveLastItem.ToolTipText = "Перейти к
последней записи";
_bindingNavigator.AddNewItem.ToolTipText = "Добавить новую
запись";
_bindingNavigator.DeleteItem.ToolTipText = "Удалить текущую
запись";
_bindingNavigator.PositionItem.ToolTipText = "Текущая позиция";
_bindingNavigator.CountItem.ToolTipText = "Общее количество
записей";


ToolStripSeparator separator = new ToolStripSeparator();
_bindingNavigator.Items.Add(separator);


_toolStripTextBoxFind = new ToolStripTextBox();
_toolStripTextBoxFind.Name = "toolStripTextBoxFind";
_bindingNavigator.Items.Add(_toolStripTextBoxFind);
```

```
_toolStripButtonFind = new ToolStripButton();
_toolStripButtonFind.Name = "toolStripButtonFind";
_toolStripButtonFind.DisplayStyle =
ToolStripItemDisplayStyle.ImageAndText;
_toolStripButtonFind.Text = "Поиск";
_toolStripButtonFind.TextAlign = ContentAlignment.MiddleRight;
_toolStripButtonFind.Image = SystemIcons.Information.ToBitmap();
_toolStripButtonFind.ImageAlign = ContentAlignment.MiddleLeft;
_toolStripButtonFind.Click += ToolStripButtonFind_Click;
_bindingNavigator.Items.Add(_toolStripButtonFind);


_toolStripButtonReset = new ToolStripButton();
_toolStripButtonReset.Name = "toolStripButtonReset";
_toolStripButtonReset.DisplayStyle =
ToolStripItemDisplayStyle.ImageAndText;
_toolStripButtonReset.Text = "Сброс";
_toolStripButtonReset.TextAlign = ContentAlignment.MiddleRight;
_toolStripButtonReset.Image = SystemIcons.Error.ToBitmap();
_toolStripButtonReset.ImageAlign = ContentAlignment.MiddleLeft;
_toolStripButtonReset.Click += ToolStripButtonReset_Click;
_bindingNavigator.Items.Add(_toolStripButtonReset);


_checkBoxFind = new CheckBox();
_checkBoxFind.Name = "checkBoxFind";
_checkBoxFind.Text = "Фильтр";
_checkBoxFind.CheckedChanged += CheckBoxFind_CheckedChanged;
ToolStripControlHost checkBoxHost = new
ToolStripControlHost(_checkBoxFind);
_bindingNavigator.Items.Add(checkBoxHost);


_bindingNavigator.AddNewItem.Click += (s, args) =>
{
    ClearInputs();
    _selectedProductAccounting = null;
```

```
};
```